

Chapter 2. Related Work

2.1 Introduction

This chapter reviews work related to this thesis. Two different areas are reviewed: IDS and immune systems. The first part of this chapter is devoted to introducing a brief taxonomy of available IDS's and introduces the two most closely related groups of IDS's: anomaly detection systems and network-based IDS's. The second part of the chapter overviews two major fields: human immune systems and AIS. Due to the complexity of human immune systems, a brief overview of human immune systems is illustrated in section 2.3. A more detailed description of human immune systems is provided in Appendix. A. Detailed Description of the Human Immune System. The rich potential of the human immune mechanism that can be applied in various areas has driven to the development of diverse algorithms under the name artificial immune systems. This is because various AIS's are modelled on particular stages of the human immune mechanism depending on their application. For this reason, this chapter reviews AIS's according to sub-components that are identified as being important for applying AIS models to IDS¹. This is followed by a summary of the fields that AIS's have been applied to. Among these areas, the AIS's that are developed particularly for computer security applications are finally presented.

2.2 Intrusion Detection Systems

An IDS is an automated system for the detection of computer system intrusions using audit trails provided by operating systems or network monitoring tools. The main goal of an IDS is to detect unauthorised use, misuse and abuse of computer systems by both system insiders and external intruders. The research on IDS's started from work by Anderson [1980], which aimed to improve the auditing facilities and the surveillance abilities of computing systems. Following this idea, the first generic intrusion detection model was proposed by Dorothy Denning in 1987 [Denning, 1987]. This generic model is independent of any particular system, application environment, and system vulnerability or intrusion type. The basic notion of the model is a real-time expert system whose knowledge is derived from statistical inference based on the audit trails of users or system resources. It stores profile facts describing the normal behaviour of subjects with respect to objects and provide the signatures of abnormal behaviours. A statistical metric and model are used to present profiles. A subject can be an individual system user, a group of system users or a system itself, while objects can be files, programs, messages, records, terminals etc. When a subject acts upon a specific object, it generates an event, which alters the statistical metric state of both subject and object. A

¹ The sub-components of the human immune system that are identified as being important for IDS are presented in Chapter 3 of this thesis.

knowledge base contains activity rules to be fired for updating profiles, detecting abnormal behaviour, and producing reports. An inference engine makes its inference by triggering rules matching profile facts. Since Denning's generic intrusion detection model was proposed, various IDS's have been developed and a number of intrusion detection systems directly employ this model [Ilgun *et al.*, 1995; Jackson *et al.*, 1991; Lunt, 1988; Liepins and Vaccaro, 1989; Lunt *et al.*, 1992; Mykerjee *et al.*, 1994].

The following sections survey the existing IDS's. As a basis, taxonomy of available IDS's is presented in the next section. Among these groups of IDS's, anomaly detection systems and network-based systems, which share many common features with immune systems, are additionally reviewed in more details.

2.2.1 Taxonomy of Intrusion Detection Systems

Currently available IDS's are categorised into several groups according to their monitoring scope and adopted detection techniques. Figure 2.1 shows this categorisation. Early intrusion detection systems operated at the *host level*, whereas contemporary systems tend to be *network-based* [Mykerjee *et al.*, 1994; Axelsson, 2000]. *Host-based IDS's* monitor a single host machine using the audit trails of a host operating system. The host-based IDS can exist in two different forms: *an intrusion detection daemon* or a *separate intrusion detection system*. The separate dedicated IDS is known to be advantageous in terms of both performance and security [Lunt, 1993]. A separate IDS avoids the performance degradation of a monitored system caused by the adoption of an intrusion detection daemon. Furthermore, it rules out the subversion of an intrusion detection daemon by the security compromise of a monitored system. Host-based IDS's can be referred to as stand-alone intrusion detection systems because their monitoring scope is restricted to only a single host in the form of a single process or a single system.

On the other hand, *network-based IDS's* [Mykerjee *et al.*, 1994; Garvey and Lunt, 1991; Habra *et al.*, 1992; Javitz and Valdez, 1991; Ko, 1996; Kumar, 1995; Paller, 1998] monitor any number of hosts on a network by scrutinising the audit trails of multiple hosts. Even though host-based IDS's have shown encouraging results [Anderson, 1993], it raises the problem of how to detect intrusions attempted across the network rather than an attempt to access only a single host. In order to detect this kind of intrusions, it is necessary for an IDS to monitor multiple events generated on several hosts to integrate sufficient evidence. Furthermore, a large proportion of computer system intrusions are achieved via intra- or inter-networks. For this reason, the use of network traffic information for security auditing renders IDS's more effective. This problem motivates the evolution from host-based IDS's to network-based IDS's

that monitor network traffic. Network-based IDS's are also referred to as distributed intrusion detection systems, because they cope with multiple hosts in a distributed environment.

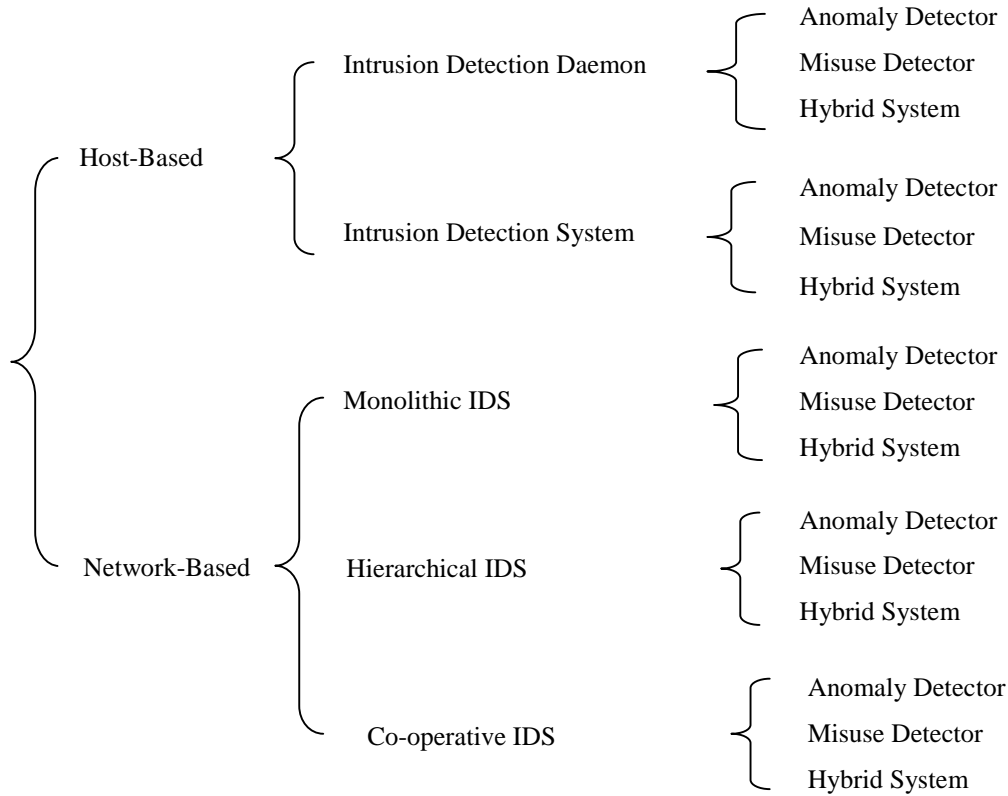


Figure 2.1. Taxonomy of IDS

Network-based IDS's can be implemented by three approaches: *monolithic*, *hierarchical* and *co-operative*. The monolithic approach is to deploy a central server to monitor multiple hosts. The rapid growth of the Internet requires security officers to monitor not only local hosts but potentially also remote hosts to which they connect. It makes it impossible to restrict the security domain to only a few hosts on a local network. Even though network-based IDS's with a central server have shown promising results in small-scale networks [Mykerjee *et al.*, 1994; Mounji *et al.*, 1995], it is difficult or impossible to support a large-scale network. Due to the number of hosts, which may be several thousand or more, a huge amount of auditing data needs to be transferred from these hosts to the central server. This causes a severe degradation of the network performance as the number of hosts grows.

The *hierarchical approach* was proposed to overcome the problems of the monolithic approach [Staniford-Chen *et al.*, 1996; Porras and Neumann, 1998]. It defines a number of hierarchical monitoring areas and each IDS monitors a single area. Instead of transferring all the collected audit data from local hosts to a central IDS, each single IDS at any level of monitoring area performs local analysis and sends its local analysis results up to the IDS at the next level in the hierarchy. The hierarchical approach seems to show better scalability by allowing local

analyses at distributed local monitoring areas. However, other problems seen in the monolithic approach still remain. When the topology of the current network is changed, it causes a change of network hierarchy and the whole mechanisms to aggregate local analysis reports must be changed [Mykerjee *et al.*, 1994]. In addition, when a monitor residing at the highest level is attacked or crashed, then all network-wide co-ordinated intrusions, which are identified only by the global analysis of local results collected from distributed monitors at lower levels, easily escape detection. The *co-operative* approach has been suggested to resolve this problem [Staniford-Chen *et al.*, 1996; Porras and Neumann, 1998; Balasubramanian *et al.*, 1998; White *et al.*, 1996; Vigna and Kemmerer, 1998]. This approach attempts to distribute a number of responsibilities of a single central server to a number of co-operative host-based IDS's. However, their ideas reported in literature are at the initial stage and the validity of the ideas has not been reported.

Host-based IDS's and Network-Based IDS's mainly employ two techniques: *anomaly detection* and *misuse detection*. The anomaly detection approach establishes the profiles of normal activities of users, systems or system resources, network traffic and services using the audit trails generated by a host operating system or a network-scanning program [Mykerjee *et al.*, 1994; Liepins and Vaccaro, 1989; Teng *et al.*, 1990; Lane and Brodley, 1997; Debar *et al.*, 1992; Obaidat and Macchiarolo, 1993; Heady *et al.*, 1983; Crosbie and Spafford, 1995b; Forrest *et al.*, 1996]. This approach detects intrusions by identifying significant deviations from the normal behaviour patterns of profiles. The assumption made in the anomaly detection approach is that if an intruder exploits the vulnerability of a system, it will show anomalous patterns in the activities of users, systems, system resources or network traffic and services. The misuse detection approach defines suspicious misuse signatures based on known system vulnerabilities and a security policy [Ilgun *et al.*, 1995; Porras, 1992; Ko, 1996; Habra *et al.*, 1992; Garvey and Lunt, 1991; Kumar, 1995; Me, 1998]. This approach probes whether these misuse signatures present or not in the auditing trails. Many host-based IDS's and network-based IDS's adopt both anomaly detection and misuse detection concurrently. This is because each technique has different strengths and drawbacks. These two components, anomaly detectors and misuse detectors, should be reciprocal in a complete intrusion detection system. However, most available commercial IDS's use only misuse detection because most developed anomaly detectors still cannot overcome the limitations described above. This trend motivates many research efforts to build effective anomaly detectors for intrusion detection purposes. The main effort of this research is also focused on the development of an anomaly detection system using an artificial immune model. For this reason, the next section discusses the major approaches used to develop anomaly detection systems for intrusion detection.

2.2.2 Anomaly Detection Systems

Various approaches have been attempted to implement anomaly detection IDS's. The basic idea of anomaly detection comprehends normal behaviours of users, systems and system resources and detects anomalous patterns that are significantly different from these behaviours. Various approaches have been attempted to implement anomaly detection IDS's. They are statistical approaches, machine learning approaches such as instance-based learning, neural networks, evolutionary computing and data mining approaches. In this section, these approaches are described with their strengths and weaknesses. In addition, the research issues related with anomaly detection systems are addressed.

2.2.2.1 Statistical Approach

Denning [Denning, 1987] suggested a number of disparate statistical approaches that are applicable to implement intrusion detection systems.

- **Operational Model:** this is the most rudimentary statistical model. The anomaly is detected merely by comparing a new observation to fixed thresholds. The thresholds are intuitively defined based on historical data. When a number of features are considered to determine the anomaly of overall system, this model is only able to probe the anomaly of an individual feature. This model is simple and easy to implement, but the interrelationships among features are ignored. This simplicity leads the system to miss sophisticated intrusions.
- **Mean and Standard Deviation Model:** this model defines the thresholds of anomalies by estimating a confidence interval. A confidence interval is a set of statistical estimates such as mean and standard deviation, which can be accepted as normal. If a new observation appears outside a confidence interval of significant level, it can be regarded as anomalous. A significance level represents the probability of wrongly determining an anomaly. The limitation of this approach is that a specific data distribution needs to be assumed when confidence intervals are defined. This often causes a practical problem in implementation, due to an unrealistic assumption of a specific data distribution.
- **Multivariate Model:** many intrusions can be detected by interrogating the interrelationship among several features concurrently. While the operational model and the mean and standard deviation models consider only specific features, the multivariate model probes a number of features collectively. It aims to elucidate the hidden meaning of correlation among features.

These statistical approaches have been adopted prevalently in the development of IDS's since Denning's initial suggestion. Haystack [Mykerjee *et al.*, 1994] is designed to help Air Force Security Officers to detect the misuses and anomalies in use of Unisys 1100/2200 computers. It uses simple descriptive statistics using mean and standard deviations. These simple descriptive statistics are employed to analyse a user's collective session behaviours and a user group's predefined acceptable behaviours. Both analyses are performed by estimating simple confidence intervals based on the Gaussian distribution.

The interrelationship between multiple monitoring targets is also often examined by making an inference from the statistical analysis of single events. In this case, IDS's equip a rule-based expert system in order to analyse several statistical results. The Multics Intrusion Detection and Alerting System (MIDAS) is such a system that has been developed for the National Computer Security Centre's networked mainframe [Mykerjee *et al.*, 1994]. The statistical profiles provide simple statistics on individual user and system activities. The rules of expert systems are used to detect anomalies by comparing the statistics in profiles, which are regarded as normal activities, with the thresholds predefined in a knowledge base, which are regarded as expected activities.

The Next Generation Real-Time Intrusion Detection Expert System (NIDES) [Lunt and Jagannathan, 1988; Javitz and Valdez, 1991; Lunt, 1993; Anderson, 1993; Jackson *et al.*, 1994] is another early anomaly detection system that adopts statistical profile-based anomaly detection. A statistical component residing on NIDES profiles the activities of individual users, remote hosts and target systems. The profiling is performed by maintaining the statistics of various measures representing these activities. It employs a multivariate method that estimates the total score showing the overall behaviour by taking into account each of the individual statistics of single measures. Furthermore, the statistics of profiles are dynamically updated using a decay rate, reflecting that the activities of most recent days contribute more than the activities of less recent days.

More advanced statistical technique is recently used by [Lane, 1999; Warrender *et al.*, 1999]. They used a Hidden Markov model (HMM) for users' UNIX command usage profiling and program behaviour profiling. HMM is a well-known statistical model for learning sequences of observed patterns and predicts the probabilities of possible outcome events. It regards a problem domain as a finite number of states and continuously updates the probabilities of possible outcome events. The probabilities of outcome events are updated by estimating their posterior probabilities, when their prior probabilities are the sequences of patterns that have been observed so far. From this work, Lane [1999] provides a fixed length of UNIX command

streams as input and considers a unique command stream as a state of HMM². When a new UNIX command is given, HMM predicts which user's input stream it is. Different work by [Warrender *et al.*, 1999] defines possible system calls of UNIX privileged processes as states of HMM. HMM models system call sequences for each privilege process and any new sequence that is not predicted by HMM is considered an anomaly. Warrender *et al.* compare HMM to three other simpler algorithms. They report that HMM shows the best accuracy on average but requires a seriously high computational cost and resource usage. More importantly, false positive error rates (that indicate the rate of mistakenly detecting normal sequences as anomaly sequences) depend more on how sequences of system calls are presented than which algorithm is used. This leads to the conclusion that research efforts should be directed more to designing how to choose input data and present it than advancing normal behaviour modelling.

2.2.2.2 Machine Learning Approach

The cornerstone of the machine learning model is to automatically extract normal activities of features, which are considered as being critical to anomaly detection, from collected audit data. When history data of features are provided, a machine learning model attempts to identify the hidden rules to define the normal behaviours of features and these identified rules are used to determine whether a newly observed event is anomalous or not.

Time-Based Inductive Machine (TIM)

Time-Based Inductive Machine (TIM) [Teng *et al.*, 1990] is a rule-based anomaly detection system. Time-based inductive learning is harnessed to generate the rules, which profile the normal temporal patterns from the observed behaviour of individual users and user groups. The rules are generated under the assumption that the sequences of events, which are snapshots of temporal processes recorded in profiles, follow a discernible pattern. For example,

$$E1 \rightarrow E2 \rightarrow E3 \Rightarrow (E4 = 95\%, E5 = 5\%)$$

where E1 through E5 are events. This rule implies that the observed sequence of events, E1->E2->E3 would result in the occurrence of E4 with the probability 95% and the occurrence of E5 with the probability 5%. During the learning phase, the rules are adjusted dynamically and finally only the rules with high confidence values remain in the system. When a new event is generated, the inductively generated rules are retrieved. The rule, whose left-hand side matches the observed sequence, is selected and the implied event according to this rule is compared to the last event in the observed sequence. If this event is deviated far from the implied event of the rule, TIM alerts the anomaly to a security officer. In addition, if a new sequence of events

² Land and Brodley [1997; 1998; 1999; 2000] also use an instance-based learning for a same problem, which is presented in section 2.2.2.2 Machine Learning Approach. More details of Lane's input data are

does not match any rule in the rule-base, this is reported to a security officer. A security officer determines whether this sequence is anomalous or a normal pattern that has not been observed before. In the case of the latter, the new sequence of events is converted into a new rule.

The problem with TIM is that it only analyses the events that occur sequentially without interruption of unrelated events. Thus, even if a legal user's tasks are recognised by TIM, when executed concurrently the unpredictable interleaving of those tasks is difficult to describe using it.

Instance-Based Learning

Another approach of machine learning for an intrusion detection system is instance-based learning (IBL) by [Lane and Brodley, 1997; 1998; 1999; 2000]. This approach profiles the input streams of an individual user and detects abnormal sequences in the input. The underlying assumption of this work is that a user responds to commands in a similar way under similar situations, and this leads to repeated sequences of actions. IBL is a learning algorithm that determines whether a new instance is close enough to previously observed instances. Thus, Lane and Brodley use IBL to determine whether a user's new input is close enough to previously observed normal inputs. If IBL classifies user's input as previously unseen, it is considered as an illegal user's attack.

In order to make this decision, IBL represents collected instances in a certain form and it calculates the similarity between a new instance and collected instances. The UNIX user input commands, which are gathered instances, are segmented into a token stream. This token stream is divided into predefined fixed length sequences and the collection of token streams is maintained in a dictionary. The novel numerical similarity measure is derived intuitively: as the number of matching adjacent tokens without separation by interleaving tokens increases, the causal relationship between two sequences of token streams becomes stronger. The original stream of similarity measures in the initial experiment is noisy due to the normal deviations in a user's actions. Lane and Brodley filter the noise out by a smoothing filter. The smoothed stream shows a steady trend indicating normality and it allows a security officer to set an appropriate threshold intuitively. As a user types a new sequence, the similarity to the sequences in a dictionary is estimated and abnormality is determined when this similarity is below the threshold.

Although this approach shows some promising results, it has a limitation with learning dynamically changing user behaviours. The training data set used for the experiments was collected over three months and seven different users participated. However, some users' usage behaviours are greatly different at the test stage from they are shown in a training data set. This causes high false positive error rates. To resolve this problem, Land and Brodley devise a token dictionary reduction method that helps IDS to contain more diverse types of user behaviours. This idea certainly contributes to keep more diverse types of user behaviours but it is difficult to assure whether it is sufficient to reduce false positive error rates to a satisfactory level. For instance, they have not tested the systems in an on-line mode, which continuously takes new user input for a long period. It implies that the effectiveness of new suggestion has not been fully investigated.

Neural Network Approach

Neural networks (NN) are renowned for their efficiency in learning [Aleksandra and Morton, 1995] and several attempts to use neural networks for anomaly detection have been made. The learning mechanism of NN is used to build profiles of normal behaviours of users, systems or systems resources. One of these attempts [Debar *et al.*, 1992] employs a recurrent NN to learn a user's regular activity. Debar *et al.* represent a user behaviour pattern in terms of a time series pattern. NN learns from a sampling window of n consecutive commands and moves this window along the time axis to obtain a new sample of n consecutive commands. After learning the time series pattern of consecutive commands, the NN predicts the next command. Each input neuron of the NN receives the user's command and after learning, the output neuron predicts the user's next command. As a result of training on 6,550 commands, the network showed up to 92.25% prediction accuracy for the first command. This experiment was performed to test the validity of a neural network as an IDS by investigating whether the prediction of a specific user's next command contributes to distinguishing intruders from authorised users.

Another attempt [Obaidat and Macchiarolo, 1993] aims to identify a user's individual keystrokes. The input patterns are comprised of the time intervals between successive keystrokes. The underlying assumption is that each person has his/her own typing technique. 15 sequential values of time intervals between keystrokes are set up as one vector to represent each user's typing technique, and 40 such vectors are collected for one user. The NN used in this experiment is a backpropagation network, which consists of 15 input nodes, 6 output nodes and 3 layers. Obaidat and Macchiarolo have verified the possibility of using NN by showing 97.8% classification accuracy in the test.

More recent work by [Ghosh *et al.*, 1999] uses a backpropagation and Elman network to monitor program behaviours, which are represented by system calls of programs. Ghosh *et al.* provides randomly generated data as “anomalous” input training data and thus allows the NN to generalise normal and abnormal program behaviours. However, this approach limits Ghosh *et al.*’s system since it requires the extra burden of creating “anomalous” data. This can be significant encumbrance especially when it has to generate sufficient data to complement dynamically changing (possibly infinite) normal data. Likewise, the success of NN in other fields has not repeated itself in the IDS field. Most work has been limited to testing the validity of its basic concept.

Evolutionary Approach

As the first attempt to use evolutionary computing (EC) for detection of network anomalies, Heady *et al.* [1991] proposed a classifier system monitoring network traffic. This system initially extracts a timestamp, packet size, source-destination address pair, and network protocol descriptor to characterise each network connection. A classifier system maintains prediction rules that are applied to one or more threshold rules to perform multivariate analysis. If all of the threshold rules fire as abnormal, then prediction rules report the network state to be abnormal. The final step of a classifier system is evolving the prediction rules using a genetic algorithm. All the prediction rules, which were created to fire in response to abnormal network states, are rated according to their prediction results. If their prediction is right, they will be rewarded, otherwise they will be penalised. A genetic algorithm uses this score as its fitness function and applies mutation and crossover operators. As the result, the weakest rules are removed and a number of new rules are generated from the remaining strongest rules using crossover and mutation.

This work shows the EC approach’s good adaptability when applied to network intrusion detection. However, Heady *et al.* [1991] reported only the initial stage of work. They did not report an evaluation of learning time, which is the critical factor for the adoption of this approach. It used only limited events and intrusions for test. It did not show how much the prediction rules evolved from initial aggregation rules could detect unknown intrusions. There are other recent works using evolutionary computing for IDS [Crosbie and Spafford, 1995b; Me, 1998, Neri, 2000]. However, these works mainly employ EC for building a misuse detector by extracting hidden intrusion signatures from intrusion data. There is no later work reported that applies EC for the purpose of anomaly detection.

2.2.2.3 Data Mining Techniques

Audit Data Analysis and Mining (ADAM) is an IDS that applies data mining techniques to discover anomalies from large amounts of network traffic data [Li, *et al.*, 2000]. ADAM employs association rule mining to summarise a large amount of given network traffic data. Association rule mining [Witten and Frank, 2000] is a data mining technique that is used prevalently to discover association relations among existing attribute values of data. ADAM uses this technique to build normal network traffic profiles. It discovers frequent events that are represented as association rules and occur in the network traffic of a specific network connection. The mined association rules are regarded as normal network traffic profiles.

Association rule-based profiles are then used to detect abnormal behaviours of network traffic for a specific network connection. In order to detect anomalies in network traffic behaviours, ADAM firstly mines association rules from newly collected network traffic data in the same way that was used for building network traffic profiles. These new rules are then compared to the association rules in the network profiles. If any association rule generated from newly collected network traffic data is not included in the network traffic profiles, this rule is considered as it indicates abnormal network traffic behaviour.

In order to detect actual intrusions, ADAM then uses a classification technique, called the Pseudo Bayes estimator [Barbara, *et al.*, 2000]. The classifier takes training network traffic data labelled with known intrusions and normal traffic. Thus, it learns network traffic patterns of known intrusions and normal network traffic. At the intrusion monitoring stage, ADAM maps suspicious association rules, that are discovered as described above, to network traffic data. These network traffic data are then presented to the classifier and the classifier decides whether these data indicate known intrusion patterns or normal traffic patterns. If they are classified as neither of these, they are considered as unknown intrusion patterns. This decision is based on the assumption that unknown intrusions are those that have not been previously observed.

ADAM is a promising approach in terms of its good scalability due to using a data mining algorithm. ADAM is indeed able to handle large amounts of messy network traffic data. However, it extracts only frequent events to build normal profiles, and this leads ADAM to detect only attacks that produce a large number of events during a short period. However, there are many intrusions that produce anomalies slowly over quite a long period. A similar approach [Lee, 1999], that uses association rule mining, has also been used to construct intrusion features from raw network data. [Lee *et al.*, 1999a] show that these features are very useful in developing a more efficient misuse detector.

2.2.2.5 Limitation of Anomaly Detection Systems

The assumption of the anomaly detection approach is that if an intruder exploits the vulnerability of a system, it will result in anomalous patterns in the activities of users, systems, or system resources. This approach detects intrusions by identifying significant deviations from the normal behaviour patterns of profiles. The strength of the anomaly detection approach is that a prior knowledge of the security flaws of the target systems is not required. Thus, it is able to detect not only known intrusions but also unknown intrusions. However, it has several limitations, which originate from its main assumption, as follows [Porrás, 1992].

- **False Negative Error:** the anomaly detection misses intrusions, which do not show anomalous behaviour. Not all intrusions are guaranteed to provide abnormal activity and this causes an intrusion detection system to falsely report absence of intrusions.
- **False Positive Error:** the anomaly detection might report legitimate users as intruders if they show behaviours that are anomalous but not intrusive. Depending on a user's personality or a special occasion, the activity of a legitimate user can be classified as anomalous. In this case, the anomaly detection approach falsely reports intrusion.
- **Gradual Misbehaviour:** If intruders are aware that their activities are monitored by the anomaly detection system, they can deceive the anomaly detector by changing their behaviours gradually. The anomaly detection system learns these changed behaviours and will judge intrusive behaviours as normal behaviours.
- **Sporadic User Environments:** the feasibility of anomaly detection approach depends on the system environment. The users in some environments such as a banking system generally show regular behaviour and access a target system enough times to build a sensible behaviour profile. However, the users in university networks do not show such regular behaviours or sufficiently frequent access to a target system. In this case, it is difficult to establish a meaningful normal behaviour profile.
- **Expensive Computation:** anomaly detection is performed based on profiles of normal behaviour. In order to maintain more sensible profiles, dynamic updating of profiles according to changes in system environment and user behaviour is necessary. This requirement brings about expensive computation.

In order to overcome these problems, the intrusion detection system should employ a misuse detection system in parallel. These two components should be reciprocal in an IDS. Hybrid

systems that adopt both an anomaly detector and a misuse detector have already been developed [Lunt *et al.*, 1992; Mykerjee *et al.*, 1994].

2.2.3 Network-Based Intrusion Detection Systems

One formidable feature of the human immune mechanism is its distributed detection. This desirable feature provides a motivation to develop a network-based IDS, and this is still one of major research issues for the intrusion detection research community. In order to scrutinise the feasibility of the artificial immune model for network-based intrusion detection, this section examines the available network-based IDS's and current research issues that have not been resolved. Network-based IDS's are developed to detect network intrusions, which attempt to subvert computer systems across the network rather than an attempt to access only a single host. Compared to single computer systems, network/distributed systems tend to be more vulnerable. This is because a network typically includes more resources to be protected, network systems are usually configured for resource sharing, and a global policy applied to all the hosts, which commonly comprise a heterogeneous set, is difficult to be defined. Moreover a network intruder is likely to attempt to intrude upon multiple points across the network [Heady *et al.*, 1983; Ko *et al.*, 1993]. The problem of protecting network systems is formidable, but unavoidable. Network-based IDS's are categorised into three groups according to overall architecture: *monolithic*, *hierarchical* or *co-operative*. In this section, the features and problems of network-based IDS's are presented according to these categories.

2.2.3.1 Monolithic Approach

The monolithic approach employs a central intrusion detection server and simple host audit programs running on multiple local hosts. Monitored local hosts transfer their collected audit trails to an intrusion detection server and then this server performs audit trail analysis. The network-based IDS's at the early days monitor only user activities, commands, application program activities or system resource activities, rather than network activities [Mykerjee *et al.*, 1994]. However, they are able to transfer the host audit trails from multiple monitored hosts to a central site for global analysis. This approach benefits from a function that converts the various formats of audit data transited from different OSs into a single canonical format, allowing the central IDS to perform global analysis.

After the addition of a simple audit trail transfer module to typical host-based IDS's, researchers' interest shifted to detection of intrusions attempted across the network. This requires auditing of network traffic simultaneously with auditing of local hosts and users. The Network Anomaly Detection and Intrusion Reporter (NADIR) and the Network Security Monitor (NSM) were the first IDS's to employ the network traffic information [Jackson *et al.*, 1991; Heberlein *et al.*, 1990; Mykerjee *et al.*, 1994]. NADIR monitors the weekly network

activity of individual users and whole network. Each local host sends raw network audit records to a single intrusion detection server for analysis. NSM models the network and hosts in a hierarchically structured Interconnected Computing Environment Model (ICEM) which has 6 layers: packet, thread, connection, host, connected network, and system layer. The lowest layer represents the network packet and the highest layer represents the state of entire network. The network traffic audited via ICEM provides input data to the expert system. This input data includes a simple network traffic summary of individual connections which each host sends for *one connection period*. The expert system analyses these input data and reports the security state of a particular connection over the time period. This is the first approach to use network connection-oriented profiling. In addition, its hierarchically structured ICEM allows NSM to adjust the granularity of analysis according to the performance of machines and the amount of traffic.

These early approaches to network-based IDS's show some critical deficiencies in their scalability, robustness and configurability. Firstly, as the network size grows, a huge number of audit trails need to be transferred from local hosts to a central server. This can cause severe degradation of the network performance and it is difficult to guarantee scalability. Secondly, if a central intrusion detection server is subverted or fails, the overall IDS becomes crippled. Thirdly, a single intrusion detection server will impose a uniform configuration despite the potentially varying requirements of each host.

2.2.3.2 Hierarchical Approach

The *hierarchical approach* was proposed to overcome the problems of the monolithic approach. It was designed to monitor large-scale networks, which have several thousand hosts. It defines a number of hierarchical monitoring areas and each IDS monitors a single area. Instead of transferring all the collected audit data from local hosts to a central IDS, each single IDS at any level of monitoring performs local analysis and sends its local analysis results up to the IDS at the next level in the hierarchy. Thus, IDS's at higher levels only need to analyse transferred local reports collectively.

The Distributed Intrusion Detection System (DIDS) developed by UC Davis, Lawrence Livermore National Laboratory, Haystack Laboratory and the US Air Force [Mykerjee *et al.*, 1994; Snapp *et al.*, 1991] is a network-based IDS using a simple hierarchy. DIDS attempted to avoid system performance degradation by delegating local analyses to local monitored hosts. The host monitor of DIDS resides on an individual host and analyses the audit records from the specific host's operating system. This performs only simple analysis and sends notable events to the central server for further analysis. In addition, the LAN monitor runs on each LAN segment and investigates all of network traffic. This architecture showed better scalability and

not to impose too many overheads on monitored hosts. It handles a fairly large number of hosts, but as the network size grows, this architecture still cannot guarantee successful scalability. This is because most of the detection decisions still depend on a central decision-maker and it limits the size of network that DIDS monitors.

Most of the network-based IDS that are currently used in a real environment employ a DIDS style architecture and audit data. They employ a central intrusion detection server, which analyses notable events generated by monitored hosts. They are Bro [Paxson, 1998], NFR (Network Flight Ranger) [Ranum *et al.*, 1997] and Shadow [Paller, 98]. These systems are freeware. Apart from these, many commercial systems such as Cisco NetRanger and ISS Real Secure [Paller, 98] also employ a similar architecture with a better graphic user interface and good summarised detection reports.

The Graph-based Intrusion Detection System (GrIDS) [Staniford-Chen *et al.*, 1996] and Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) [Porras and Neumann, 1998] are IDS's that use a hierarchical approach in a more sophisticated way. GrIDS uses network graphs showing network activities and these graphs are used to identify large-scale of network attacks. GrIDS views the whole of a large network as the aggregation of a number of sub-networks. A data source module executing at any host monitors host and network activity and it sends reports of detected anomalous activity to a graph engine. These reports are converted into a graph by the graph engine and this graph shows the network activity of each sub-network. The network activity graphs are then passed up to the graph engine of another sub-network in a higher level. The graph engine in the higher level in turn builds graphs that have a coarser resolution. The graphs at high levels show only summarised information about lower level graphs rather than showing a complicated topology of the combined network. This hierarchical approach makes GrIDS scalable. Another hierarchical network-based IDS, EMERALD [Porras and Neumann, 1998], was developed in order to monitor a large enterprise networks with thousands of users connected in independent administrative domains. The network surveillance is achieved through three hierarchical layered analyses: service, domain and enterprise-wide. On the service analysis level, EMERALD handles the misuse of individual components and network services within a single domain. On the domain analysis level, it detects misuses visible across multiple network services and components. On the enterprise-wide level, it monitors co-ordinated misuses across multiple domains. This hierarchical analysis is achieved through execution of service monitors, domain monitors and enterprise-wide monitors respectively on local hosts, the entry points of independently administered sub-network and gateways. The detection decisions made by each monitor are disseminated to other monitors at any level for global analysis.

The hierarchical approach seems to show better scalability by allowing local analyses at distributed monitoring areas. However, other problems present with the monolithic approach still remain. When the topology of the current network is changed, it causes a change of network hierarchy and the whole mechanism for aggregation of local analysis reports must be changed [Mykerjee *et al.*, 1994]. In addition, when a monitor residing at the highest level is attacked or crashes, then all network-wide co-ordinated intrusions, which are identified only by the global analysis of local results collected from distributed monitors at lower levels, easily escape detection.

2.2.3.3 Co-operative Approach

The *co-operative* approach attempts to distribute the responsibilities of a single central server to a number of co-operative host-based IDS's. Each IDS is responsible for monitoring only a small aspect of a local host and a number of IDS's operate concurrently and co-operate with each other. Moreover, they can make a coherent inference and global decision. The difference of this approach from the hierarchical approach is that there is no hierarchy amongst the distributed local IDS's. Therefore, the failure and subversion of any one IDS does not always prevent the detection of co-ordinated attacks.

Co-operative security managers (CSM) [White *et al.*, 1996] is an example of a co-operative IDS approach. It comprises two distinct components: a distributed intrusion detection component and an intruder tracking handling component. The distributed intrusion detection component co-ordinates local decisions made by local intrusion detection components and the intruder tracking component finds the origin of an intruder through the communications of CSMs on other hosts. However, the detailed mechanism of each component has not been reported yet following the initial proposal. Autonomous Agents for Intrusion Detection (AAFID) developed by [Balasubramanian *et al.*, 1998; Crosbie and Spafford, 1995a; Crosbie and Spafford, 1995b] is the first attempt to use autonomous agents for network intrusion detection. Each independent autonomous agent is responsible for monitoring only a small aspect of the overall system and a number of agents operate concurrently. Through the co-operation among them, AAFID can make a coherent inference and finally a global decision. Currently, the second prototype of AAFID is released and this implements only the communication mechanism between components and hosts. Therefore, it has not been proved yet that the overall architecture actually supports the presented research claims.

In these works, it is claimed that most of problems encountered by the other two approaches previously mentioned would be resolved. The prototypes developed in these projects have not implemented the full cooperative functionality presented in the proposed model, and thus the validity of this claim remains unproven. In particular, this approach raises a different problem,

namely efficiency. It places many overheads on monitored local hosts such as many communication mechanisms, auditing mechanisms and analyses of audit trails, and these can be a significant encumbrance to them.

2.3 Overview of Human Immune Systems

This section introduces a brief overview of human immune systems. The overview presented in this section is largely based on [Paul, 1993; Tizard, 1995]. A more detailed description of human immune systems is presented in appendix A. Detailed Description of the Human Immune System.

The overall human immune system is implemented through the interactions between a large number of different types of innate and acquired cells rather than the function of one particular human organ. Among a large number of different cells, lymphocytes (white blood cells) play a central role. Their main function is to distinguish self cells, which are the cells of human body, from non-self cells, which are dangerous foreign cells. Each lymphocyte is specialised in reacting to a limited number of structurally related cells, known as antigens. Lymphocytes have specific binding areas, called receptors, which have complementary shapes to the determinants of antigens, called epitopes. A specific antigen is recognised by its epitopes binding to lymphocyte antibody receptors. More details about specific antigen recognition by lymphocytes are presented in appendix A.1.1. Specific Recognition.

Lymphocytes are classified into two main types: B-cells and T-cells. B-cells are antibody-secreting cells and T-cells kill antigens or help or suppress the development of B-cells. Antibodies are proteins and they play the part of receptor of B-cells. Both B-cells and T-cells have their own unique genetic structures. Both the genes of B-cells and T-cells are expressed by several chains of DNA (*gene libraries*) and each chain has a variable domain and a constant domain. The genes in the variable domain are highly variable from one to another and this determines the specific binding area to antigens. The genes in the constant domain are invariable and show various biological effects when B-cell antibody receptors bind to antigen epitopes. Appendix A. 1. 2. Genetic Structure of Lymphocytes provides more information about genetic structures of B-cells and T-cells.

B-cells and T-cells are developed in the bone marrow and the thymus respectively. At the bone marrow and the thymus, several gene libraries uniquely corresponding to domains of B-cells and T-cells contain the candidate genes to express B-cell and T-cell receptors. A specific receptor is generated by selecting gene segments randomly from gene libraries and joining them. Furthermore, in order to generate diverse receptors, they adopt a progressive series of genetic operators during their development processes. These include gene rearrangements, choosing different joining sites, somatic mutation, class switching and others. The details of

these genetic operators with the B-cell and T-cell development process are presented in appendix A.1.3 Development.

Before leaving the bone marrow and the thymus, maturing B-cells and T-cells have to pass the last test, negative selection. In B-cell and T-cell development process, totally new cell receptors can be generated via various genetic operators. Therefore, it leaves the possibility for randomly generated receptors to bind to self cell epitopes. To prevent this, when maturing B-cells and T-cells bind to self cells circulating through the bone marrow and the thymus, they are killed instead of being released into the body. Figure. 2.2 (left) shows the development of B-cells and T-cells in the bone marrow and the thymus. The figure A.3 in appendix A.1.3 provides details about the process of gene rearrangement shown in figure 2.2 (left).

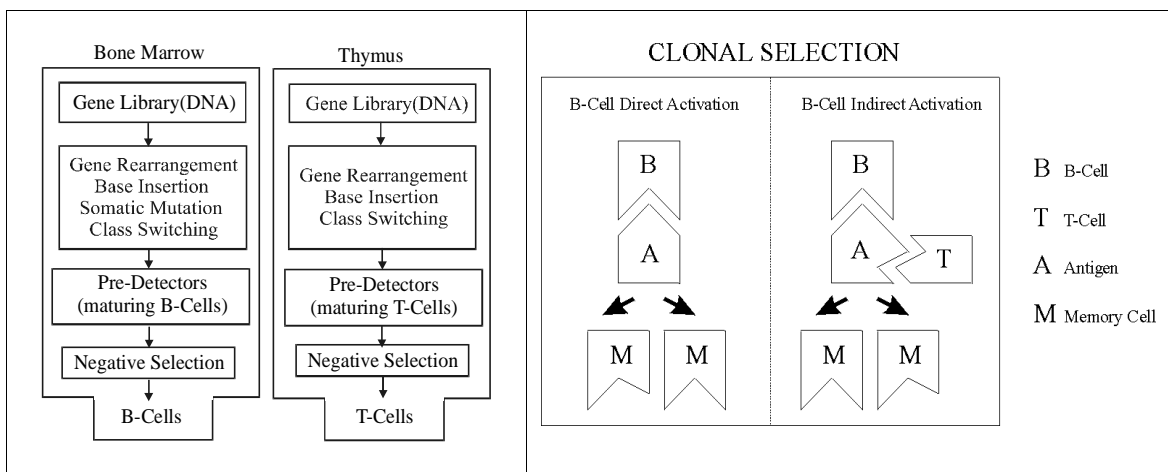


Figure 2.2 Development of B-cells and T-cells (left). Clonal selection (right)

Mature B-cells and T-cells that pass the negative selection are released from the bone marrow and thymus. Both B-cells and T-cells continuously circulate around the body in the blood and encounter antigens for activation and evolution. The antibodies of B-cells, which recognise harmful antigens by binding to them, are activated directly or indirectly. When B-cell antibody receptors bind to antigen epitopes with strong affinity above a threshold, they are directly activated. On the other hand, B-cell antibody receptors can bind to antigen epitopes with weak affinity below a threshold. In this case, B-cells need the help of T-cells and Major-Histocompatibility Complex (MHC) molecules to be activated. MHC molecules have two important functions to help B-cell activation. Firstly, they bind to the fragments of antigens specially hidden inside cells (not visible on the cell surface), and secondly they transport these fragments to the B-cell surface. When B-cell antibody receptors bind to antigen epitopes with weak affinity, MHC molecules try to find some hidden antigens inside cells. When MHC molecules find them, they transport them on the surface of B-cells. The receptors of T-cells are genetically structured to recognise the MHC molecule on the B-cell surface. Thus, T-cells can bind to MHC molecules on B-cell surfaces. When the T-cell binds to an MHC molecule with strong affinity, it sends a chemical signal to the B-cell which allows it to activate, grow and

differentiate.

Then, what makes the T-cells determine B-cell activation? One major difference between B-cells and T-cells is that only B-cells perform somatic mutation, which is a very high rate of mutation, increasing their diversity when they develop in the bone marrow. Hence, B-cells have more various and new receptors than T-cells. In addition, the thymus is centrally located while the bone marrow is distributed. Thus, most self-cells pass through the thymus and hence the negative selection in the thymus is more reliable than that in the bone marrow. Therefore, the final decision of B-cell activation with weak affinity is made by the T-cells. To summarise, when a B-cell binds to an antigen with strong affinity above a threshold, it directly causes this B-cell to activate, grow and differentiate. On the other hand, if a B-cell binds to an antigen below a threshold with weak affinity, it needs the assistance of helper T-cells to be activated. Accordingly, different antigens, which are not same but similar enough, can be detected by the approximate binding of a single antibody. More detailed information about B-cell and T-cell activation can be found in appendix A.1.4 Activation. In addition, obtaining self-tolerance through negative selection is also presented in appendix A.1.6 Self Tolerance.

With or without the assistance of T-cells, B-cells activate and this activation is immediately followed by clonal selection. As one species evolves via natural selection, human immune systems also evolve through a process called *clonal selection*. The genes, which determine the specificity of antibodies, continuously evolve toward having the capacity to detect more prevalent antigens at any moment and reproduce new lymphocytes with high affinity for those specific antigens. When B-cells are activated, they are divided and differentiate into a number of clones of antibody secreting cells. These clones can have the same antigen-binding properties as the receptors of parent B-cells or they can have mutated antigen-binding properties. As B-cells bind more to the specific antigens that are currently prevalent, they have more chance of being selected for cloning. Similarly, as they bind less to these specific antigens, they have fewer chances to be selected to clone and eventually tend to decrease. Since antigens constantly change, the efficiency of detection is maintained by the evolution of B-cell antibodies via clonal selection. Furthermore, when antigens activate B-cells, they produce memory cells for the reoccurrence of the same antigens in the future. Due to these memory cells, antigens that have been identified previously are detected much more quickly (this is known as the secondary response). Figure 2.2 (right) shows the generation of memory cells via clonal selection. More information about clonal selection is provided in appendix A.1.5 Evolution.

In contrast, idiotype antibodies, which are anti-antibodies, can activate antibody receptors. Immune systems let antigens and anti-antibodies compete in binding to antibodies, and the winning anti-antibodies can suppress binding between antigen and antibody. The inhibition of idiotype antibody against antigen contributes to the regulation of an appropriate level of immune responses. Jerne, an immunologist, proposed the immune network theory [Jerne, 1974] based on

understanding the role of the idiotype antibody. He views immune systems as a functional network of lymphocytes, and the network at any moment has a dynamic state of internal interactions of antibodies and antigens. The continuous chain of differentiation by antigens and suppression by idiotype antibodies can form a large-scaled network. When this network finally reaches the equilibrium status between suppression and stimulation, it determines the overall immune system. This theory is an abstract model to illustrate how overall immune responses can be self-regulated. Nevertheless, there are still many controversies over whether this theory indeed describes how the overall immune responses are regulated or not [Callard, 2000]. Appendix A.1.6 Self Tolerance describes the self-regulation of immune network theory in depth.

2.4 Artificial Immune Systems

As introduced in the previous section, the biological immune system has been successful at protecting the human body against a vast variety of foreign pathogens. In the last few years, a growing number of computer scientists have carefully studied the success of this competent natural mechanism and proposed artificial immune models for solving various problems [Dasgupta, 1998a; De Castro and Von Zuben, 1999]. Various approaches to build AIS have been attempted mainly by implementing a small subset of overall human immune mechanisms [Dasgupta, 1998a; De Castro and Von Zuben, 1999]. This section provides a review of these systems categorised according to which mechanisms of human immune systems are implemented. In addition, this section briefly highlights the application areas where AIS have been applied. More details about various AIS and application areas are provided in later chapters.

2.4.1 Negative Selection Algorithm

The human immune system employs negative selection to eliminate immature antibodies, which bind to self cells passing by the thymus and the bone marrow. Of the newly generated antibodies, only those which do not bind to any self cell are released from the thymus and the bone marrow and distribute throughout the whole human body to monitor other living cells. This stage of the human immune system is important to ensure that the generated antibodies do not to attack self cells [Percus *et al.*, 1993; Tizard, 1995].

[Forrest *et al.*, 1994; Forrest *et al.*, 1997] proposed a negative selection algorithm mimicking this process. They consider a negative selection process of the human immune system as a sophisticated anomaly detection process. This process does not define specific harmful cells to be detected and thus allows the human immune system to be able to detect previously unseen harmful cells. This algorithm consists of three phases: defining self, generating detectors and monitoring the occurrence of anomalies. In the first phase, it defines ‘self’ in the same way that other anomaly detection approaches establish the normal behaviour patterns of a monitored system. It regards the profiled normal patterns as ‘self’ patterns. In the second phase, it generates a number of random patterns that are compared to each self pattern defined in the

first phase. If any randomly generated pattern matches a self pattern, this pattern fails to become a detector and thus it is removed. Otherwise, it becomes a ‘detector’ pattern and monitors subsequent profiled patterns of the monitored system. During the monitoring stage, if a ‘detector’ pattern matches any newly profiled pattern, it is then considered that new anomaly must have occurred in the monitored system.

This negative selection algorithm has been successfully applied to detect computer viruses [Forrest *et al.*, 1994; 1996], tool breakage detection and time-series anomaly detection [Dasgupta, 1996; 1998b]. Besides these practical results, D’haeseleer *et al.* [1997] theoretically analysed several advantages of negative selection as a novel distributed anomaly detection approach.

This thesis also employs the negative selection algorithm to detect network traffic anomalies. More detailed features and drawbacks of this algorithm are mainly discussed in chapter 4. The extension of this algorithm by adding other human immune mechanisms is also studied throughout this thesis. In addition, other computer security related systems employing this algorithm are briefly introduced in section 2.5 Artificial Immune Systems for Computer Security.

2.4.2 Clonal Selection Algorithm

The human immune system learns dynamically changing antigens via clonal selection. Activating antibodies are divided into a number of clones that have the same antigen-binding properties as parent B-cells, or mutated antigen-binding properties. On the other hand, if an antigen cannot activate B-cells within a limited time, it rapidly dies off. Therefore, based on the existing antigens, only the fittest B-cell antibodies survive. Since antigens constantly change, the efficiency of detection is maintained by the evolution of B-cell antibodies via clonal selection [Tizard, 1995].

De Castro and Von Zuben [2000a] propose a clonal selection algorithm, named CLONALG, that directly mimics this process. CLONALG consists of antigen and antibody populations. An antigen population stores a training data set and an antibody population has candidate solutions. At every generation, CLONALG clones n selected antibodies proportionally to their fitness: the higher the antibody fitness, the higher number of clones. The cloned antibodies are mutated and the mutation rate is also determined depending on antibody fitness: the higher the fitness, the lower the mutation rate. From the mutated clones, CLONALG selects m mutants having highest fitness and these mutants are competing with the originally selected n antibodies. Only k winners remain in an antibody population and l antibodies having lowest fitness are replaced by random antibodies. Likewise, CLONALG adopts several features of clonal selection in human immune systems: the selection and cloning of the most stimulated antibodies, death of non-stimulated antibodies, applying mutation on clones and re-selection of clones proportionally to their fitness. This algorithm follows a very similar concept to evolutionary

algorithms (EA) in terms of employment of fitness-based selection and applying a genetic operator to produce variation. The distinct feature of this algorithm from other EA is the method of determining an optimal antibody cloning rate and a mutation rate. Nevertheless, De Castro and Von Zuben [2000a] did not report possible effects of this new feature on performance.

While CLONALG borrows a clonal selection idea for an EA-like algorithm, other work [Hunt and Cooke, 1996; Hunt *et al.*, 1998; Mitsumoto *et al.*, 1997; Watanabe *et al.*, 1998a, b; Ishiguro, 1997] embed this feature in the immune network model³. The common feature of these AIS is implementation of clonal selection via the selection and cloning of the most stimulated antibodies and death of non-stimulated antibodies. This feature is often considered to be similar to EA, which is also based on natural selection.

An augmented GA approach proposed by [Forrest *et al.*, 1993] applies the clonal selection idea to AIS. Forrest *et al.* pay attention to a niching strategy of the clonal selection process and borrows this strategy to maintain generality and diversity of antibodies used for GA. They explored whether the niching strategy of clonal selection is able to i) detect common patterns of randomly presented antigens and ii) to maintain the diverse antigen population. In their model, they used the GA to evolve the antibody population under a constant antigen population. Mirroring the niching strategy of the human immune system, for each generation, their modified GA selects an arbitrary size of random sample from the antibody population, and a single random antigen from the antigen population. After each antibody in the sample is matched against a selected antigen, the fitness score of the one highest-scoring antibody is increased while the fitness scores of the others remain the same. They compared this niching strategy of the artificial immune system with the fitness sharing algorithm [Smith *et al.*, 1993]. From this comparison, they reported that as the result of antibody sampling mechanism, the niching strategy of the AIS controls its generality via the antibody sample size. This immune-based niching strategy is later applied successfully to dynamic scheduling [Hart *et al.*, 1998; Hart and Ross, 1999]. This thesis studies application of this immune based niching strategy for building an efficient misuse detector in chapter 5.

2.4.3 Gene Library Evolution

The human immune system learns dynamically changing antigens via clonal selection. As the result of clonal selection, the surviving antibodies and memory cells are able to detect various non-self antigens they are exposed to during their life-times. However, it is known that the genes that define surviving antibodies and memory cells cannot be directly written back on the DNA (which is a gene library) of an egg or sperm cell. Consequently, the genes that define surviving antibodies are not passed on to the next generation of the immune system [Paul, 1993; Sompayrac, 1999]. However, it is also known that the learning results via clonal

³ The review of immune network model is presented in the section 2.4.4 Immune Network Theory.

selection with hypermutation during a lifetime indirectly lead the evolution of a gene library in the human immune system over generations. Although the genes of useful antibody mutants are not directly inherited, individuals that had more useful mutants are more likely to survive against various types of pathogens. Thus, the gene libraries of these individuals are passed over generations and offspring having these inherited gene libraries are more likely to have an immune system with a good *capability* of producing useful mutants. This effect was firstly proposed by Mark Baldwin in 1896 and named as the Baldwin effect [Baldwin, 1896].

Since the Baldwin effect was proposed, many researchers have attempted to understand the relation between learning and evolution in nature. Hightower *et al.* [1996] have reported that “learning accelerates evolution” by simulating the gene library evolution of the human immune system. They used GA to let the gene library of a binary immune system evolve. Other work by Hightower *et al.* [1995] investigated what determines the gene library evolution’s direction. (i.e. where the selection pressure of gene library evolution is headed). This question is about what the evolution strategy of the human immune system is, when a vast number of dynamically changing antigens needs to be covered by a much smaller number of antibodies. This work showed that the binary antibodies of a binary immune system evolve toward an equilibrium between maximum coverage of antigen space and least overlapping coverage of antibody space.

Oprea and Forrest [1998; 1999] advanced Hightower *et al.*’s work [1995] further and studied the diversity required of a gene library in the human immune system and the role of gene library evolution. This work reported that the gene library tends to evolve towards the point of mapping mainly antigens that have experienced so far. The subsequent study by the same author [Oprea, 1999] investigated the role of hypermutation by investigating its mutating targets. Her experiments showed that hypermutation usually attempts to mutate the antigen-binding regions from a gene library, and the mutation results often led to fine-tuning of antigen-binding parts.

There are two methods employed by the currently available AIS’s in order to make their gene libraries evolve. The first approach directs gene library evolution through a Baldwin effect [Hart and Ross, 1999; Michaud *et al.*, 2001; Fukuda *et al.*, 1998b; Watanabe *et al.*, 1998a, b; Ishiguro *et al.*, 1997; Lee *et al.*, 1999b; Ishida, 1996a; 1997; Watanabe and Ishida, 2002] and the second approach allows for the provision of direct feedback from learning results to a gene library [Dasgupta *et al.*, 1999a; Hart *et al.*, 1998; Gaspar and Collard, 1999; Hajela and Yoo, 1999; Potter and De Jong, 1998; Nikolaev *et al.*, 1999; Timmis, 2001; Fukuda *et al.*, 1998a]. These two different approaches have been implemented in various ways depending on the adopted AIS model. The details of these works are discussed in depth in chapter 7 of this thesis.

Whilst there seems to be no particular distinction between learning using clonal selection and the evolution of gene library for AIS, research in the evolutionary computation field has been working more actively on the topic of the relation between learning and evolution [Nolfi and Floreano, 1999]. This thesis also examines the application of gene library evolution for the proposed integrated AIS in chapter 7.

2.4.4 Immune Network Theory

Jerne's immune network theory [Jerne, 1974] describes that antibody-antibody binding triggers stimulating the proliferation of antibody clones as well as the suppression of antibodies. The continuous chain of differentiation by antigens and suppression by idiotypic antibodies can form a large-scale network. When this network finally reaches an equilibrium state between suppression and stimulation, it determines the overall immune system.

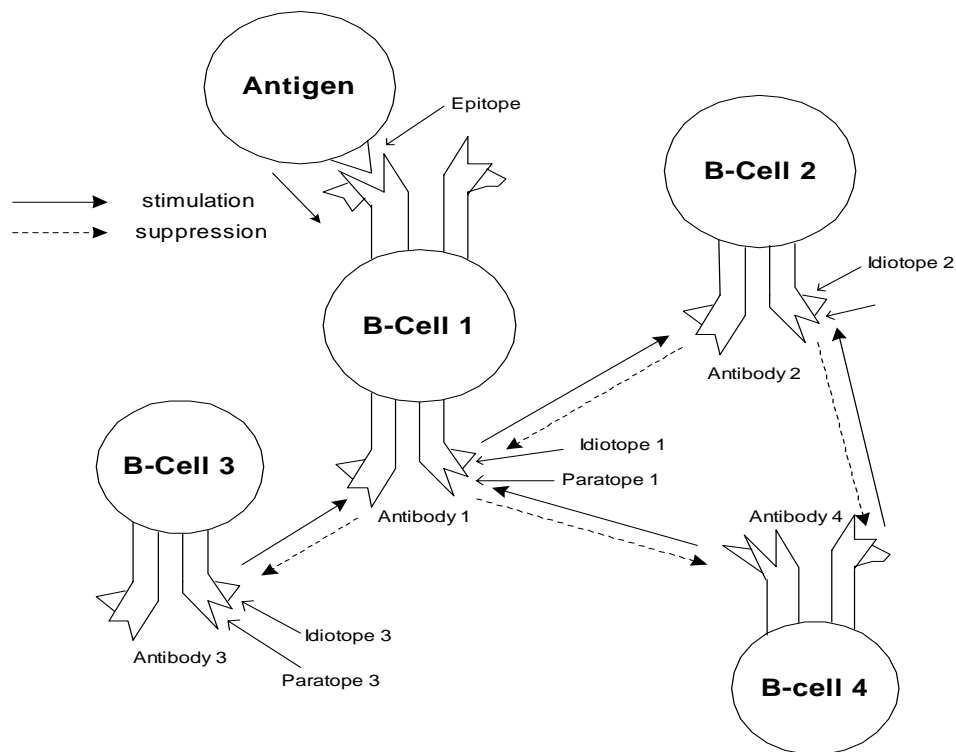


Figure 2.3. Jerne's Immune Network, [Timmis, 2001]

This theory particularly emphasises the parallel and distributed nature of immune systems. It addresses that the millions of local immune responses between a single antibody and a single antigen, or a single different antibody, occur in parallel and in disparate locations, and these interactions lead to the final appropriate level of immune response. In addition, another significant assumption of this theory is that the overall immune response is regulated not only by the interactions between lymphocytes and antigens but also the interactions among the lymphocytes themselves, even in the absence of antigens. This theory is an abstract model that illustrates how the overall immune response can be self-regulating. Nevertheless, there is still

much controversy over whether or not this theory really describes how the overall immune response is regulated or not [Callard, 2000].

Despite this controversy, various cognitive scientists and computer scientists [Farmer *et al.*, 1986; Varela *et al.*, 1988] have examined Jerne's Immune Network theory to understand the dynamics of human immune systems and devise a novel learning algorithm. Among them, Farmer, Packard and Pearson [Farmer *et al.*, 1986] are pioneers who interpreted Jerne's Immune Network theory in terms of a new computational intelligence paradigm. They specified Jerne's rather abstract model using binary strings to represent antibodies and antigens. They formalised the concentration level of each antibody, determining the number of antibodies produced in cloning. The formula that determined the concentration level of each antibody embedded four factors: i) the stimulation by existing matching antigens, ii) the stimulation by neighbouring antibodies, iii) the suppression by neighbouring antibodies and iv) the tendency of cell to die due to a finite life span. It should be noted that the neighbouring antibodies could both stimulate and suppress a given antibody. More precisely, as shown in figure 2.3, each lymphocyte has two different types of protein determinants: paratopes and idiotopes. Antibodies treat any folded protein (located on the surface of a cell) that binds to paratopes as an antigen determinant. In contrast, they consider a folded protein binding to idiotopes as an antibody determinant. Regardless of what has matched with paratopes and idiotopes, the concentration level of a given B-cell is boosted by the affinities of paratopes and reduced by the affinities of idiotopes. Thus, if paratopes of a current antibody bind paratopes of other neighbouring antibodies, the concentration level of a given antibody is increased. Similarly, it decreases if paratopes of a current antibody bind the idiotopes of other neighbouring antibodies.

More importantly these pioneers interpreted this model as a John Holland's Classifier system [Holland, 1975], which was already well-known as a computational intelligence paradigm. This interpretation triggered many computer and cognitive scientists to investigate use of an artificial immune model for computational intelligence purposes. Among them, work by Varela *et al.* [1988] elucidated that the immune system is cognitive because it has four significant features. They are i) recognition of other molecules, ii) memory of encountered molecules, iii) determination between self and non-self molecules and iv) associative memory to recognise molecules that have not been encountered. Then, they showed how these features could emerge by eliciting three different immune network properties: *structure*, *dynamics*, and *metadynamics*. The immune network *structure* is the network connection pattern. As Jerne's theory pointed out, this pattern was made by interactions between encountered antigens and existing lymphocytes. This mechanism verified that immune recognition formed the network structure and that network structure described the memory kept in the given immune system. The immune network *dynamics* represent the dynamic changes of a network structure according to newly encountered molecules. The network structure could change because updated affinities of lymphocytes to new molecules (whether they are antigens or other lymphocytes) make new connections and delete

existing connections. Finally, the immune network *metadynamics* referred to another type of change of network structure but by addition and deletion of network nodes. While the immune network *dynamics* are changes in network structure due to the affinity updates (equivalent to connection weight changes), the immune network *metadynamics* emphasise the continuous adoption and deletion of molecules in the immune system. The following work formulised these three properties and these formulas were used to solve an adaptive control problem in further work [Bersini, 1991; Bersini & Valea, 1994].

2.4.5 Immune Memory

Immunologists define *immune memory* as the capability of the immune system to fully protect the body from attack from pathogens that have previously been detected [Yates and Callard, 2001]. When B-cells activate, they produce memory cells to protect against the reoccurrence of the same antigens. Memory cells enable the immune system's response to previously encountered antigens (known as the *secondary response*), which is known to be more efficient and faster than non-memory cells' response to new antigens [Tizard, 1995; Paul, 1993]. These cells are long-lived, often lasting for many years or even for the lifetime of an individual [Tizard, 1995].

AIS's have implemented the immune memory of the human immune system in various ways. Among the implementations, the immune network theory has been the particularly popularly used [Timmis, 2001; Bersini and Varela, 1994; Manderick, 1994]. Studies of this mechanism verified that immune recognition formed the network structure and this network structure described the memory of the immune system. The new network formed by surviving B-cells is expected to provide a new solution to a newly appearing environment, without losing the solutions to the previous environment. This is possible because the AIS selects surviving B-cells in the network not only by their antigen stimulation level but also by antibody suppression level. Although some antibody secreting B-cells did not receive a sufficient degree of stimulation from new antigens, they would not be deleted as long as they did not also receive a high degree of suppression from other antibodies. The B-cells having these antibodies remain and act as memory cells in the AIS.

In addition, some other work [Gaspar and Collard, 1999; Nikolaev *et al.* 1999] induces artificial immune memory by modifying a fitness function of a Genetic Algorithm (GA). The lack of maintenance of diversity is a well-known problem of GA. A simple GA often lets a solution population evolve toward one optimal point and has some difficulties with handling a multi-modal function. In order to resolve this problem, [Gaspar and Collard, 1999; Nikolaev *et al.* 1999] adds the antibody-antibody suppression concept of the immune network theory to the fitness function of GA. Gaspar and Collard [1999] adds an *endogenic activation* fitness function that indicates the similarity to other antibodies in the population. Gaspar and Collard's GA selects the best antibodies that are solutions of a current optimisation target. This GA also selects other antibodies that are quite dissimilar to currently selected best antibodies. Different work by

[Nikolaev *et al.*, 1999] devises a new dynamic fitness function using Genetic Programming (GP). The new GP dynamic fitness function takes into account whether each individual matches more important examples and different individuals match the examples in different niches. The overall degree of proliferation of a given individual was measured by the total number of matching examples weighted by the importance of each example. The dynamic fitness function also considers the extent of suppression of a specific individual to be defined by the difference between a given individual's and other individuals' matching examples.

Another type of immune memory employed for AIS's is Sparse Distributed Memory (SDM) [Smith *et al.*, 1996]. Smith *et al.* [1996] viewed that the approximated recalling mechanism of SDM is equivalent to the primary and secondary responses of memory cells. Consequently, Smith *et al.* [1996] claimed that immune memory is robust and associative, as is SDM, since both employ a similar approximate addressing mechanism. Hart and Ross [2001] adopted SDM in their co-evolutionary GA to cluster moving data. In their work, the SDM was used for antibody and antigen matching and recalling. It lets each antibody vote (=i.e. match and recall) several antigens instead of one antigen. Thus, when a new antigen is presented, the voting result from all antibodies decides the label of the antigen, whether self or non-self.

In contrast to above approaches, Hofmeyr's AIS [1999] adopts a separate memory cell population that is isolated from other antibody populations. This system is the only AIS to implement immune memory by directly mimicking memory cells of the human immune system. It has explicit antibodies to retain memory of previously detected antigens and these antibodies are treated differently from other antibodies. They remain in a population for a longer period than other antibodies and they detect antigens much more quickly than other antibodies.

The AIS developed in this thesis also adopts immune memory and more detailed issues about immune memory are studied in chapter 7.

2.4.6 Artificial Immune System Applications

This section briefly introduces application areas where AIS have been applied. Various application areas utilise AIS's, mainly in order to achieve autonomous, dynamic and distributed features of a system. These areas are as follows:

- Computer Security [Forrest *et al.*, 1997; Hofmeyr, 1999; Hofmeyr and Forrest, 2000; Libicki, 1997; Dasgupta *et al.*, 1999b; Okamoto and Ishida, 1999; Stillerman *et al.*, 1999; Somayaji and Forrest, 2000; Kephart, 1994; Kephart, *et al.*, 1997; 1998; Narasimhan *et al.*, 1999; Lamont, *et al.*, 1999; Williams, *et al.*, 2001; Gu *et al.*, 2000]
- Computer System Maintenance [Burgess, 1998a; b; 2000]
- Concept Learning [Potter and De Jong, 1998; Hunt and Cooke, 1996; Hunt *et al.*, 1998; Nikolaev *et al.*, 1999]

- Data Clustering [Timmis, 2001; De Castro and Von Zuben, 2001a]
- Robot Control [Ishiguro *et al.*, 1997, Watanabe *et al.*, 1998a; 1998b; Lee *et al.*, 1999b]
- Fault Detection [Ishida, 1996a; 1997; Dasgupta and Forrest, 1996; Watanabe and Ishida, 2002]
- Optimisation [Hajela and Yoo, 1999; Fukuda *et al.*, 1998a; De Castro and Von Zuben, 2000a; Toma, *et al.*, 1999; Gaspar and Collard, 1999; 2000]
- Scheduling [Mori *et al.*, 1998; Hart *et al.*, 1998; Hart and Ross, 1999]
- Aircraft Control [KrishnaKumar and Neidhoefer, 1998]
- Production Line Management [Fukuda *et al.*, 1998b]
- Pattern Recognition [Smith *et al.*, 1993; McCoy and Devarajan, 1997; De Castro and Von Zuben, 2000a]
- Spectra Recognition [Dasgupta *et al.*, 1999a]
- Vaccine Design [Smith *et al.*, 1998; 1999]
- Multi-Agent Systems [Dilger, 1997; Ishida, 1996b; Dasgupta, 1998b; Ballet *et al.*, 1997; 2000]
- Open Web Server Policy Management [Suzuki and Yamamoto, 2000; 2001]
- Protein Structure Prediction [Michaud *et al.*, 2001]
- Fault Tolerant System Design [Bradley and Tyrrell, 2000a, 2000b; 2001]

As seen in the previous sections, existing AIS's have diverse forms and each different form is based on different parts of the human immune system. This diversity allows a rather wide range of application areas to use AIS for different purposes. The application area that this thesis investigates is computer security. The next section discusses in depth the study of the application of AIS's to computer security problems.

2.5 Artificial Immune Systems for Computer Security

Computer security is one of the application areas that AIS have been most frequently applied to. The similarity between the goal of a computer security system and the human immune system naturally leads researchers to examine analogies between the two. This section introduces three of the most studied different computer security related applications that adopt human immune features.

2.5.1 Virus Detection

Since computer viruses have been identified as a destructive form of artificial life [Spafford, 1994], it is very natural for computer scientists to investigate the human immune system in order to understand its formidable defence mechanism against harmful biological viruses. Stephanie Forrest at University of New Mexico is a pioneer in the study of the application of human immune system techniques to protection of computer systems from destructive artificial life. Forrest *et al.* [1994] view virus detection as a “self-nonsel discrimination problem” in a computer. They regard monitoring targets (such as legal user activities, legal application usage

activities, uncorrupted data, etc.) as self and expect the AIS to discriminate them from others (such as illegal user activities, illegal application usage activities, virus infected data, etc.). This models the negative selection process of human immune systems (this model is introduced as a negative selection algorithm in section 2.4.1 Negative Selection Algorithm [Forrest *et al.*, 1994; 1996]). In their work, Forrest *et al.* generates detectors from a standard binary executable .com file and then tests whether generated detectors can detect a virus infected .com file. The experiment results show that the AIS obtained 100% detection rate with 125 detectors when an infected file is encoded by 655 binary strings each string having 32 bits. However, their work has not been further tested against other viruses, of which there are recently new varieties.

Another recent approach called Computer Virus Immune System (CVIS) [Lamont, *et al.*, 1999; Harmer and Lamont, 2000; Harmer, 2000], by the US Air Force Institute of Technology team, attempts to apply the latest AIS of Stephanie Forrest's team at University of New Mexico. The Forrest team's latest AIS [Hofmeyr, 1999; Hofmeyr and Forrest, 2000] is developed to detect a certain set of network intrusions and is not tested for virus detection. CVIS employs the negative selection algorithm with some novel ideas that were used in [Hofmeyr, 1999; Hofmeyr and Forrest, 2000] such as life span, activation threshold and costimulation. As new features, CVIS adds detected virus analysis, repair of infected files and dissemination of analysis results to other local systems. In addition, CVIS is designed to operate under a distributed environment using autonomous agents.

Although they design the AIS under a distributed environment, the reported test results of CVIS [Harmer, 2000] are limited to evaluation of whether CVIS detects various viruses on a local host. The tested viruses are the "TIMID" virus, which infects .COM files only within a local directory, and viruses included in the standard test file provided by the European Institute for Computer Anti-Virus Research. The test reports show the sensitivity of detection and error results depending on different matching thresholds. By setting appropriate parameters, their system is able to show a detection rate of up to 89%. One serious problem found from these tests is scalability. It requires about 1.05 years for generated antibodies to scan an 8GB hard disk drive. Harmer [2000] also studies various negative selection algorithm matching functions for the first time.

For a virus detection system that can operate under a distributed and heterogeneous environment, Okamoto and Ishida [1999] proposes a multi-agent based AIS. This system consists of four different types of agents: antibody, killer, copy and control agents. Antibody agents detect viruses from local hosts and killer agents neutralise viruses by overwriting self information on infected files. Copy agents copy uninfected files, which are equivalent to ones infected, from different hosts and control agents helps communication among other agents. This system is clearly limited by their repair method since there are only limited numbers of files that have equivalent ones from other hosts.

A different approach to using AIS for virus detection is undertaken by [Kephart *et al.*, 94; 97; 98; White *et al.*, 2000] at the IBM research centre. While above approaches model a particular sub-mechanism of human immune systems as a novel algorithm, the AIS developed by [Kephart *et al.* 94; 97; 98] identifies some traits of the human immune system that make it attractive for virus detection purposes and implements them using established algorithms. Thus, Kephart *et al.* design their AIS with the following five stages, each inspired by the human immune system: i) detect a previously unknown virus on a user's computer, ii) capture a sample of the detected virus and send it to a central computer, iii) analyse the virus sample automatically and derive an instruction for detecting and removing it from any computer host, iv) send the derived instruction to the user's computer, adding it into the anti-virus data files used by the anti-virus software, and run the anti-virus product to detect and remove all occurrences of the virus, and v) disseminate the derived instruction to other computers in the user's locale and to the rest of the world. From these five stages, for instance, Kephart *et al.* [94; 98] claims that the first stage of their AIS, which detects a previously unknown virus on a user's computer, is a useful trait of the innate human immune system. Since the innate human system provides a non-specific immune response, Kephart *et al.* view the detection of previously unknown viruses in a generic way as the equivalent task in their artificial system. The actual implementation of this innate immune response is carried out by two established algorithms: generic disinfection techniques and neural networks, which are used to build a generic classifier [Kephart, *et al.*, 1997]. Similarly, the other four stages of the AIS are also understood comparable in effect with some sub-procedures of the human immune system, but their implementation is completed using other more conventional algorithms.

The problem that this system particularly addresses is the speed of analysis of infected files and dissemination of the analysed virus signatures with removal instructions to other computer systems [White *et al.*, 2000]. As the speed at which computer viruses spread becomes alarmingly faster, this problem is becoming more significant than ever in attempting to reduce damage to information resources. The prototype by Kephart *et al.* [94; 98] is developed, eventually into a commercial system in [White *et al.*, 2000], and this commercial system benefits from the idea of sending killing signals by immune cells. As the human immune system clones useful antibodies and these spread rapidly through the body in order to counter the speed of harmful antigen spread, the key feature of White *et al.*'s computer immune system is that it disseminates virus signatures quickly to a large number of computer systems across the world. In order to achieve this, they set up a network that connects computers within monitoring domains to a centralised analysis centre. Any host within this domain sends files that are infected or potentially infected to the central analysis centre, which extracts unknown virus signatures from them. The newly found virus signatures are always sent, not only to the sender of infected files, but also to any computer within the monitoring domain. In order to

complete this process rapidly, the virus analysis process at the centre is fully automatic, requiring no human involvement.

This commercially used system [White *et al.*, 2000] developed at the IBM research centre delivers somewhat different messages from Stephanie Forrest's team's work. The IBM research team attempts to identify and understand useful processes of the human immune system, and to see how these can help with devising a new virus detection product. However, they do not attempt to implement the processes using the mechanism of the human immune system, only to mimic it at a high level of abstraction. They advance other conventional algorithms to implement identified human immune processes. In other words, the IBM team treats each process of the human immune system as a black box and thus the actual implementation of this box is not considered important to provide the desired result from each process. This may have assisted them in building a commercially successful system.

2.5.2 Intrusion Detection

The first attempt to use an AIS for intrusion detection is the host-based IDS [Forrest *et al.*, 1996; 1997; Hofmeyr *et al.*, 1998]. A host-based IDS using AIS profiles the normal behaviours of privileged system processes and compares them to currently observed behaviours. This first computer immunology approach for intrusion detection does not employ many features of human immune systems, such as distributed anomaly detection. Instead, it tries to use the basic concept of the human immune system: detection of non-self entities. This is not much different from the conventional anomaly detection mechanism. An interesting point of their work is that it shows that system call sequences of privileged process can be a good indicator, describing the self of a monitored host. More importantly, their work shows that the profiling of system calls can result in a massive reduction of audit data and thus this approach can help to build a light-weight IDS. However, it needs to be tested on more extensive intrusions in order to prove whether this level of audit is good enough to replace other more extensive audit levels, such as auditing all application programs activities, auditing all typed user commands and auditing system resources and network traffic activities.

This early attempt encouraged the same research team to extend AIS to network intrusions. Forrest *et al.* [1997] proposes the application of computer immune systems to computer security for the first time. From their work, they argue that the analogy between human immune systems and artificial immune systems works best for network intrusion detection problems. Similarly, the work in this thesis [Kim and Bentley, 1999a; 1999b] has identified the features of human immune systems that can contribute to the development of effective network-based IDS. This analysis will be presented in the next chapter in more detail. More recent work by [Hofmeyr, 1999; Hofmeyr and Forrest, 2000] develops an AIS for network intrusion detection, called LYSIS. LYSIS implements the AIS proposed in [Forrest *et al.*, 1997]. It employs various features of the human immune system such as activation threshold,

life span, memory detectors, costimulation, etc., in order for it to monitor dynamic self and non-self behaviours. In addition, LYSIS operates under a distributed environment and is tested to monitor 50 hosts in a local area network (LAN). The system considers the typical network traffic paths between two hosts as self and detects abnormal network packet paths. In order to perform this, LYSIS extracts a “datapath triple”, which is a source host IP address, a destination host IP address and a TCP service (port) number from TCIP/IP packet headers. This “datapath” is used as input data to build self-profiles. LYSIS is tested over seven intrusions and shows promising results. However, this rather limited input data requires future research to evaluate whether LYSIS is able to detect more diverse intrusions than those used in [Hofmeyr, 1999; Hofmeyr and Forrest, 2000]. The AIS studied in this thesis shares many features of LYSIS and provides new understanding of these features that were not reported by [Hofmeyr, 1999; Hofmeyr and Forrest, 2000].

Different work inspired by LYSIS has recently been reported in [Williams *et al.*, 2001]. Williams *et al.* extend Computer Virus Immune Systems (CVIS) reported at [Lamont, *et al.*, 1999; Harmer and Lamont, 2000] to detect network intrusions. This extended system, called Computer Defense Immune System (CDIS), also uses the negative selection algorithm with some novel ideas that were introduced in LYSIS such as life span, activation threshold, and costimulation. In contrast to LYSIS, CDIS chooses 28 features of TCP packet headers and 16 features of UDP and ICMP packet headers as its input self data. For antibody string generation, CDIS randomly selects the network protocol and chooses between two and seven features of that selected protocol. For the selected protocol features, CDIS generates the values of these features randomly. As a new way of reducing the computing time to generate antibody strings, CDIS adopts an affinity maturation process. This process aims to optimise each antibody to cover the non-self space as much as possible without matching any self string. In order to perform this, CDIS uses a genetic algorithm and its fitness function is defined as the growth rate of non-self space coverage by each antibody. The performance of CDIS is tested over two data sets: the first set with 20K self packets and the second set with 1.3M self packets and both data sets are collected during one day. For a non-self data set, Williams *et al.* simulate various intrusions on attack-free network traffic data. The test results show that CDIS is able to detect simulated intrusions without serious self detection errors. It also verifies that the costimulation and affinity maturation help CDIS to reduce both false positive and false negative error rates. However, the affinity maturation requires far too much computation time to be applied to the second, larger, data set. In addition, the authors also point out that the high detection rates with low error rates may possibly be obtained because the simulated intrusions are limited. More extensive tests with more varied intrusions are required.

Dasgupta *et al.* [1999b] proposes a general framework for Immunity-Based IDS. This framework applies a multi-agent architecture to support an AIS model for intrusion detection. This immunity-based IDS framework simply follows the multi-level detection feature of

human immune systems rather than employing more sophisticated features of human immune systems, such as distributed detection, clonal selection, negative selection, etc. The multi-agents residing within this model monitor systems at various levels in a hierarchical manner, activate warning signals, communicate their local warning signals and make decisions based on collected local warning signals. In order for the proposed IDS to reduce false warnings, this model emphasises the importance of collective decisions. It monitors anomalies of users' system usage patterns, system resource usage patterns, process activity patterns, and network traffic patterns at the same time and produces final warning signals only when correlation among local warnings is proven to be intrusion-related. The ART-2 NN is employed to detect anomalies of all monitoring levels and fuzzy logic was proposed to combine four different levels of warnings into a final threat warning [Dasgupta and Brian, 2001]. The different work [Dasgupta *et al.*, 2000] instead uses the Classifier system to draw final collective decisions. The fitness function used for the Classifier system is defined by three factors: 'How well each rule follows security expert's heuristic rules', 'the diversity of a rule set', and 'the generality of a rule'. [Dasgupta *et al.*, 2000; Dasgupta and Gonzalez, 2001a] reports some preliminary results related to the performance of the Classifier system as a global decision maker.

Since the negative selection algorithm was proposed by Forrest *et al.* [1994] as a novel anomaly detection algorithm, many researchers have been inspired by this pioneering work. However, at the same time, many researchers have been suffering from the innate problems of this algorithm. Dasgupta and Gonzalez [2001b] have examined whether the fundamental idea of negative selection is advantageous for a the network intrusion detection problem. They have compared a negative characterisation approach to a positive characterisation one. The positive characterisation approach is one of conventional anomaly detection algorithms that usually define normal profiles and detect anomalies by monitoring a significantly deviated event. A self set is a collection of vectors that present one event observed at time t and these vectors are grouped into a number of sets. When a new data is given, this approach searches for a nearest self vector to a given data point and measures the Euclidean distance between these two points. If this distance is larger than a pre-defined threshold value, a given datum is considered anomalous. On the other hand, their negative characterisation approach employed a genetic algorithm in order to generate detector rules covering niches of a non-self space. While the negative selection algorithm generates detectors covering the same radiuses of clusters in a non-self space, Dasgupta and Gonzalez [2001b] uses the GA to let detectors evolve to cover generalised niches that have various radiuses. In order for detector rules to evolve, the fitness function was defined by the volume of non-self space covered by detector rules after a penalty having been applied according to the number of matching self examples.

The experimental results in their work show that the negative characterisation approach is more efficient than the positive characterisation one in terms of space required. However, this claim is drawn from a rather unfair comparison. This is because the self profiles used for the positive

characterisation approach are not represented by their generalised patterns, while the detector rules for the negative characterisation approach are chosen ones, generalised by the GA. In addition, it is shown that positive characterisation outperforms in terms of its detection rate. However, their work shows no comparison between these two approaches with respect to computing time, which has been revealed as a severe problem of a negative selection algorithm from this thesis [Kim and Bentley, 2001a] (see chapter 4. Negative Selection Algorithm). In addition, distributed detection using a negative characterisation approach is not investigated either, even though this is regarded as a potential strength of negative selection over positive selection.

Stillerman et al., [1999] introduce an immunity-based intrusion detection approach that is particularly applicable to Common Object Request Broker Architecture (CORBA) applications. CORBA is a popular common messaging middle-ware that enables the communication of distributed objects comprising a distributed application. Their work focuses on detection of attacks on CORBA applications, termed rogue client attacks. This attack is of the “legal user’s misuse” type. Its approach to detecting this type of attack is closely related to the early AIS reported in [Forrest et al., 1996; 1997; Hofmeyr et al., 1998], in that it is not much different from conventional anomaly detection. As in [Forrest et al., 1996; 1997; Hofmeyr et al., 1998], a self database is built using a sliding time window to define a self pattern from collected sequences of requested methods. After sufficient self-patterns have been collected, new patterns that are not found in the self database are regarded as anomalous. To reduce false positive error rates in the system, it waits until a sufficient number of anomalies from the same client have been observed. The experimental results show that the system is able to detect anomalies caused by this attack without high false positive error rates. Although this report shows that their work is feasible when applied to the CORBA application level of attack, the report does not include exact false positive error rates or the training and test data collection periods in the experimental results. Furthermore, their work does not utilise any of the new ideas of AIS instead using a conventional anomaly detection approach called an immune-based system.

2.5.3 Fault Tolerant Software

Strictly speaking, fault tolerant computer systems are not in the computer security application field. However, the research goals of this area are similar to those of computer security research. A fault tolerant computer system attempts to assure reliable operation of a system despite presence of hardware or software flaws. This section introduces two ongoing pieces of research concerning the development of fault tolerant hardware and software using the AIS concept.

In an effort to build fault tolerant software, Burgess [Burgess, 1998a; 1998b; 2000] develops a proactive computer system maintenance program using an immunity approach. A modern computer environment changes rapidly and often a network domain is heterogeneous. Despite

such a dynamic environment, the configurations of computer systems are often maintained manually by users. Furthermore, in a local network environment, a system administrator may force a centralised policy on a group of local users without considering various local conditions, which will inevitably require different optimal configuration rules. These problems require new thinking in order to build a dynamic and distributed software maintenance program. For the development of this new type of program, Burgess [1998a] addresses some useful features of human immune systems: autonomous and distributed detection of harmful antigens and damaged cells, and a healing process to sustain a healthy status of a body. These features are already understood to be useful attributes of other security-related systems. A great deal of effort in implementing these features for computer systems has been spent on the self and non-self distinction model, achieved using a negative selection algorithm. However, Burgess [1998a] claims that the self and non-self distinction concept is too simple to explain the whole human immune mechanism and thus too straightforward to be applied for AIS. Instead, he advocates a different immunology theory called a danger model. Although this model is still controversial among immunologists, he considers that this model seems to be more appropriate to AIS. The single attribute that makes this model different from other immune theories is that immune response is triggered by unusual deaths of self cells. Following this model, Burgess [1998a] puts the emphasis of computer immune system on an autonomous and distributed feedback and healing mechanism, triggered when a small amount of damage can be detected at an initial attacking stage.

The cornerstone of Burgess's AIS is a program called *cfengine*, a tool that builds an expert system for maintaining configuration policies of UNIX network machines [Burgess, 1998b; 2000]. After a human administrator initially specifies configuration policies at a very general level using an expert system shell, *cfengine* automatically monitors the state of each system and adapts initially specified policies to be more locally optimised. This change immediately triggers the modification of other policies affecting different hosts. The new policy reflects a new environment and other hosts can optimise their own policies. The stages of this feedback loop can be summarised as, firstly, analysis of the current state of each host, detection of any noticeable change of observed state, altering the corresponding configuration policy in order to heal this change if the observed change shows negative effects, disseminating an updated local policy to other hosts, and triggering local policies of other hosts receiving the updated policy. Burgess [1998a] argues that this feedback loop follows the operation of the most significant immune mechanism, which is autonomous and distributed feedback and healing, triggered by detection of a small amount of damage at an initial attacking stage. Similar to a virus detection system developed by the IBM research team [Kephart et al. 1994; 1998], Burgess's AIS follows a similar philosophy. The relevance to the study of AIS is their understanding of useful immune mechanisms, and the implementation of these mechanisms is not their main concern. Thus, conventional algorithms are used for each process in the AIS healing/feedback loop.

The second work introduced in this section is research concerning the development of hardware fault tolerant systems using an AIS approach [Bradley and Tyrrell, 2000a; 2000b; 2001]. Bradley and Tyrrell use a negative selection algorithm and a new approach is taken to represent self and non-self sets. They use a finite state machine (FSM) representation to define a valid self set. The FSM defines all acceptable states of a hardware system and existing transitions between them. The main motivation for using a FSM to represent a self set is for minimisation of the search space where antibodies search for matches with antigens. Bradley and Tyrrell test a negative selection algorithm on a rather small self set, consisting of 40 self strings each of 10 bit length. This is created from two user inputs: count enable and reset, with 4-bit current state and a 4-bit subsequent state. The experimental results are evaluated in terms of the detection rate according to different matching thresholds. All the experimental results are consistent with the conclusion other earlier work by [D'haeseleer, 1997; Forrest *et al.*, 1994]: the number of antibodies generated greatly affects the detection rate.

2.4 Summary

This chapter has reviewed two different areas that are relevant to the an artificial immune model for intrusion detection. Firstly, the basic mechanism of IDS and a brief taxonomy of existing IDS's are presented. From the IDS categories stated in this taxonomy, anomaly detection systems and network-based IDS's are discussed in particular depth. Despite the advantages of these two approaches for intrusion detection, the difficulties of applying these two approaches hinder the development of effective IDS's that adopt them. For anomaly detection systems, three main approaches have been used: statistical, machine learning and data mining. However, all of these approaches have suffered from similar problems originating from the inherent characteristics of anomaly detection. Network-based IDS's are also categorised into three groups according to their overall architecture. These three groups are monolithic, hierarchical and co-operative. These are introduced and the features and limitations of each approach are also outlined.

The second part of this chapter has briefly introduced the overall human immune mechanism and extensively surveyed existing AIS's. In the last few years, a growing number of computer scientists have carefully studied the success of this natural mechanism and proposed artificial immune models for solving various problems. However, due to the complexity of the overall human immune mechanisms, AIS's have mainly implemented a small subset of the overall human immune mechanisms. Among these AIS's, five of the most commonly used artificial immune algorithms are introduced. The common observation from these algorithms is that none of these systems fully provides the remarkable features that are integral to the human immune system. This observation is also found in the computer security application area. One group of AIS's used for a computer security application [Forrest *et al.*, 1994; 1996; Hofmeyr, 1999; Hofmeyr and Foresst, 2000; Harmer, 2000; Dasgupta *et al.*, 1999b; Stillerman *et al.*, 1999; Bradley and Tyrrell, 2000a; 2000b; 2001] mainly use a small subset of overall human immune mechanisms and thus their performance has only been verified in very limited test

cases. Another group of AIS's is used for a computer security application [Kephart et al, 1994; 1998; White et al., 2000; Williams et al., 2001; Burgess, 1998a; 1998b; 2000] is commercially used or tested on a real environment. However, these AIS's only follow the basic concepts of the human immune mechanisms and actual implementation use available conventional algorithms.

The overall conclusion is that it is necessary to propose a new artificial immune model that directly employs some features of the human immune system, but can still achieve the efficiency needed in a realistic application environment. Such a system is proposed in the next chapter and shows how this new model can provide benefits to an IDS in a network environment.